Casey Grage and Renee Zha

Checkpoint 5

Use TensorFlow to implement the neural network models for your project. Describe what you learned from this analysis. Was the property you were trying to predict easy or hard to model? Did you have to experiment with different features in order to get your results?

Modeling with Neural Networks:

1. Predict the identity group of an officer given a TRR or complaint report (or the identity group of the victim if applicable) by training the model on the text of complaints and TRRs from different identity groups

(link to databricks notebook: https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/3752198143680027/3643970726338521/1677315359957412/latest.html)

For the corpus of text, we decided to use the "narratives" column from the cases_case table in the settlements database. These short descriptions provide a story for each case from a seemingly journalistic point of view, making them ideal for performing text analysis on. However, the sample size is not very large (only around ~900 total cases).

We implemented a multi-layer perceptron (MLP) model using unigram and bigram tokenizations to represent each narrative (with heavy guidance from this resource: https://developers.google.com/machine-learning/guides/text-classification/) to solve this classification problem. Because of the small sample size, we decided to classify only based on the officer's race, and not by their gender as well. Doing so would have probably resulted in too few representatives in certain categories (American Indian and Asian/Pacific females especially), though the distribution of race itself is already quite unbalanced and likely affected the model's accuracy.

Initial results of MLP model with 2 64-unit intermediate layers:

```
Train on 2345 samples, validate on 261 samples
Epoch 1/1000
 - 2s - loss: 1.5307 - acc: 0.5313 - val_loss: 1.4255 - val_acc: 0.5441
Epoch 2/1000
 - 1s - loss: 1.3064 - acc: 0.6077 - val_loss: 1.2312 - val_acc: 0.5556
Epoch 3/1000
 - 1s - loss: 1.1076 - acc: 0.5923 - val_loss: 1.1093 - val_acc: 0.5556
Epoch 4/1000
 - 1s - loss: 0.9846 - acc: 0.6141 - val_loss: 1.0512 - val_acc: 0.5709
Epoch 5/1000
 - 1s - loss: 0.9043 - acc: 0.6499 - val_loss: 1.0217 - val_acc: 0.5747
Epoch 6/1000
```

```
 - 1s - loss: 0.8419 - acc: 0.6832 - val_loss: 1.0028 - val_acc: 0.6015
Epoch 7/1000
 - 1s - loss: 0.7943 - acc: 0.6989 - val_loss: 0.9904 - val_acc: 0.6054
Epoch 8/1000
 - 1s - loss: 0.7521 - acc: 0.7087 - val_loss: 0.9934 - val_acc: 0.6054
Epoch 9/1000
 - 1s - loss: 0.7214 - acc: 0.7164 - val_loss: 0.9928 - val_acc: 0.6169

Model evaluation:

 32/652 [>.............................] - ETA: 0s
192/652 [=======>......................] - ETA: 0s
384/652 [================>.............] - ETA: 0s
640/652 [============================>.] - ETA: 0s
652/652 [==============================] - 0s 295us/step
Accuracy: 0.596625766505
```

After attempts to tune hyperparameters (4 32-unit intermediate layers):

```
Train on 2345 samples, validate on 261 samples
Epoch 1/1000
 - 5s - loss: 1.5910 - acc: 0.4725 - val_loss: 1.5588 - val_acc: 0.5556
Epoch 2/1000
 - 2s - loss: 1.5066 - acc: 0.5753 - val_loss: 1.4301 - val_acc: 0.5556
Epoch 3/1000
 - 2s - loss: 1.3030 - acc: 0.5795 - val_loss: 1.1679 - val_acc: 0.5556
Epoch 4/1000
 - 2s - loss: 1.0886 - acc: 0.5872 - val_loss: 1.0919 - val_acc: 0.5556
Epoch 5/1000
 - 2s - loss: 1.0060 - acc: 0.6000 - val_loss: 1.0392 - val_acc: 0.5594
Epoch 6/1000
 - 2s - loss: 0.9208 - acc: 0.6307 - val_loss: 1.0047 - val_acc: 0.5900
Epoch 7/1000
 - 2s - loss: 0.8527 - acc: 0.6742 - val_loss: 0.9951 - val_acc: 0.6207
Epoch 8/1000
 - 2s - loss: 0.8071 - acc: 0.6968 - val_loss: 1.0063 - val_acc: 0.6169
Epoch 9/1000
 - 1s - loss: 0.7690 - acc: 0.7126 - val_loss: 1.0146 - val_acc: 0.6245

Model evaluation:

 32/652 [>.............................] - ETA: 0s
192/652 [=======>......................] - ETA: 0s
384/652 [================>.............] - ETA: 0s
640/652 [============================>.] - ETA: 0s
652/652 [==============================] - 0s 265us/step
Accuracy: 0.596625766505
```

Despite changing around parameters including # of layers, # of units, and learning rate of the model, the test accuracy seemed to stay around 59-60%. Considering that the percentage of white officers in this particular dataset is also around 58%, it is tough to say that this model is much better at predicting an officer's race from just guessing based on demographic populations. Again, most likely limitations are due to the fact that the corpus of data was not very large, and more investigation needs to be done with other sources of text data (complaint reports?). In addition, more experimentation could be performed on different types of models (the MLP model was suggested based on the metrics of the data, however models that rely on sequence tokenizations could also be investigated). In particular, it would be interesting to see if the "sentiment" of the narratives change substantially based on the identity groups of the people involved.